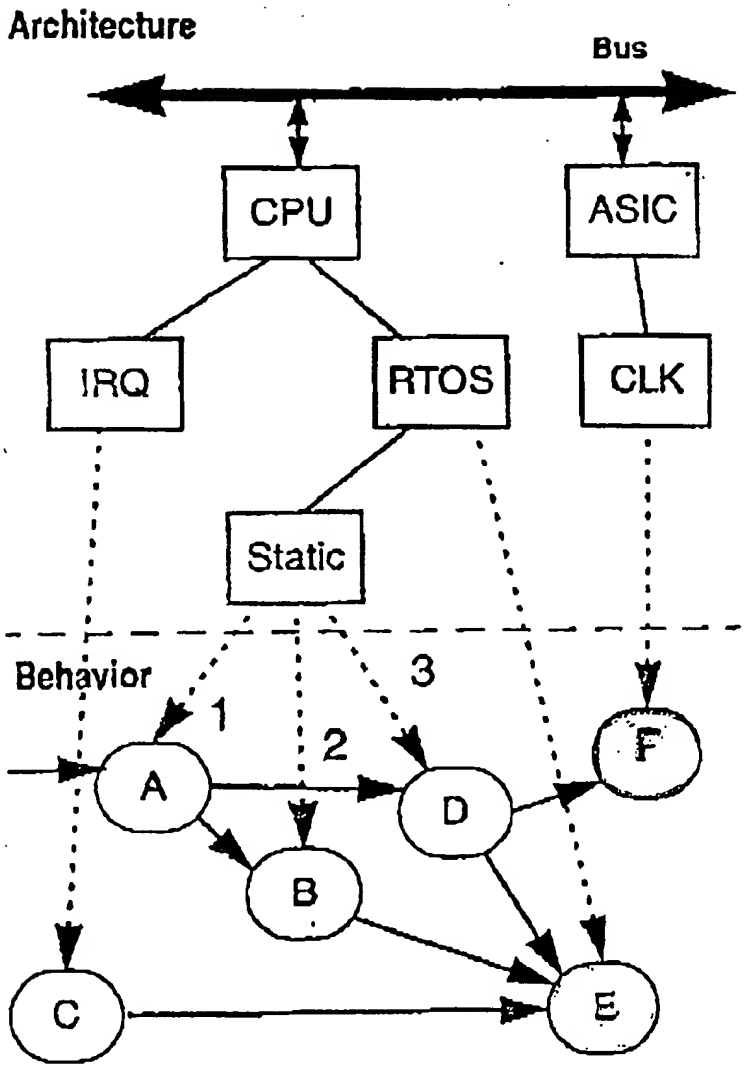
Architecture
Bus
CPU
ASIC
IRQ
RTOS
CLK
Static
Behavior
1
2
3
A
B
C
D
E
F

FIG. 1

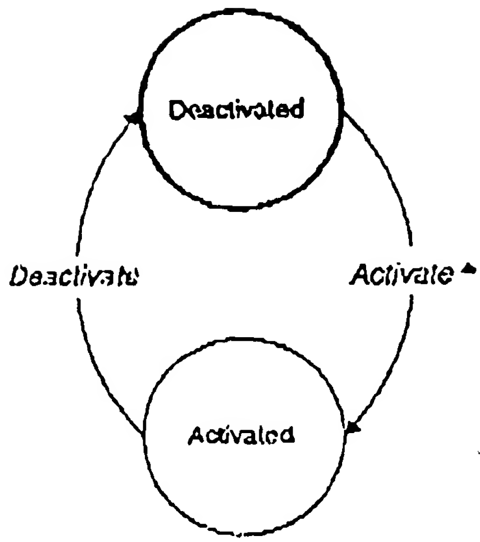

FIG. 2a

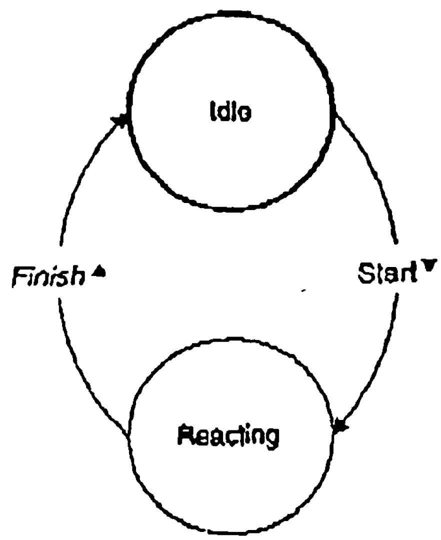

FIG. 2b

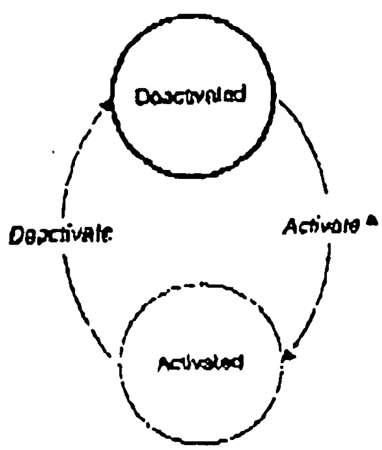

FIG. 3a

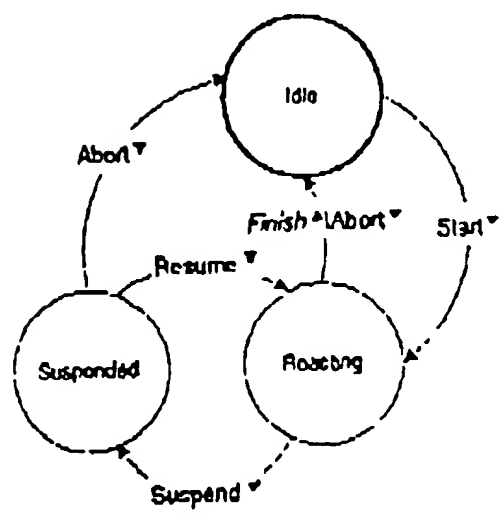

FIG. 3b

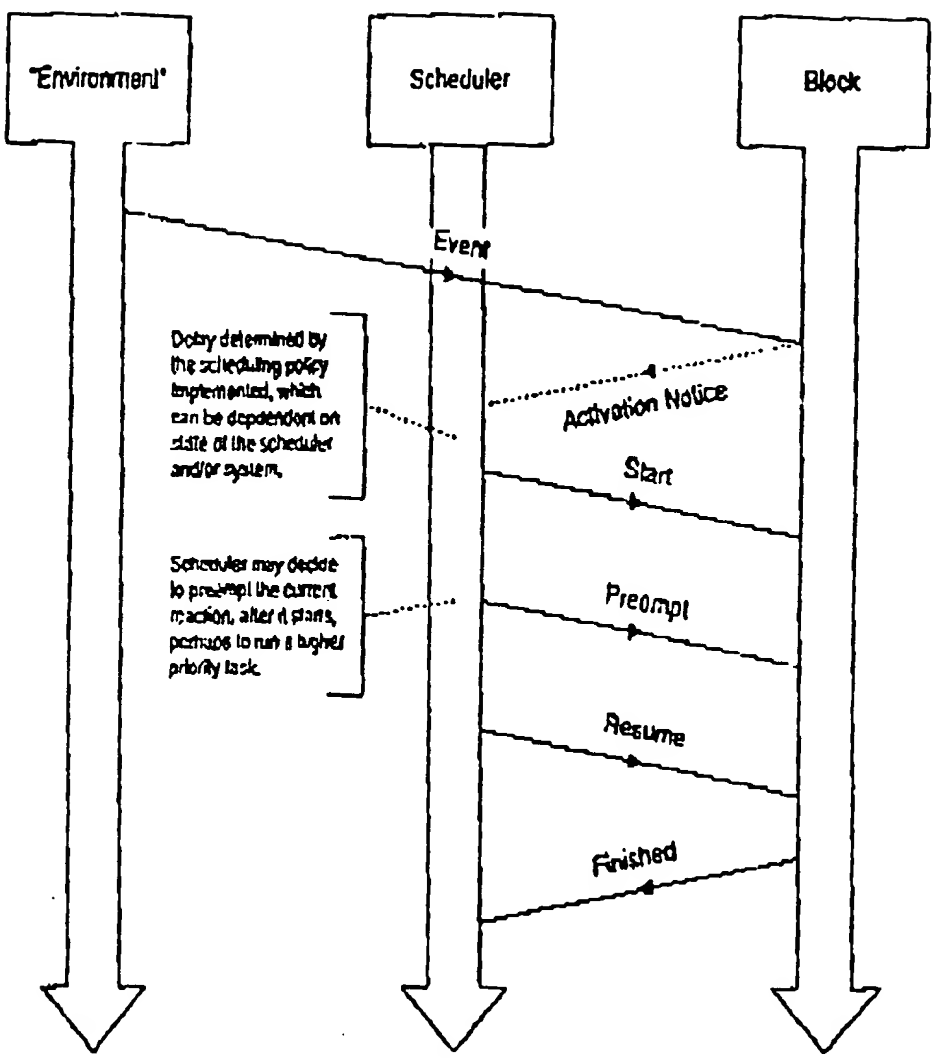
"Environment"
Scheduler
Block
Event
Delay determined by the scheduling policy implemented, which can be dependent on state of the scheduler and/or system.
Activation Notice
Start
Scheduler may decide to preempt the current reaction, after it starts, perhaps to run a higher priority task.
Preempt
Resume
Finished

FIG. 4

```
class FxSchedulerInterface
{
public:
        //
        // Messages sent from FxSchedulableInterface objects.
        virtual void schedulerActivationNotice(
                FxSchedulableInterface *) = 0;
        virtual void schedulerFinishNotice(
                FxSchedulableInterface *) = 0;
};
```

FIG. 5

```
class FxSchedulableInterface
{
public:
        //
        // Messages sent to FxSchedulerInterface object.
        virtual void schedulableStart() = 0;
        virtual void schedulableSuspend() = 0;
        virtual void schedulableResume() = 0;
        virtual void schedulableAbort() = 0;
};
```

FIG. 6

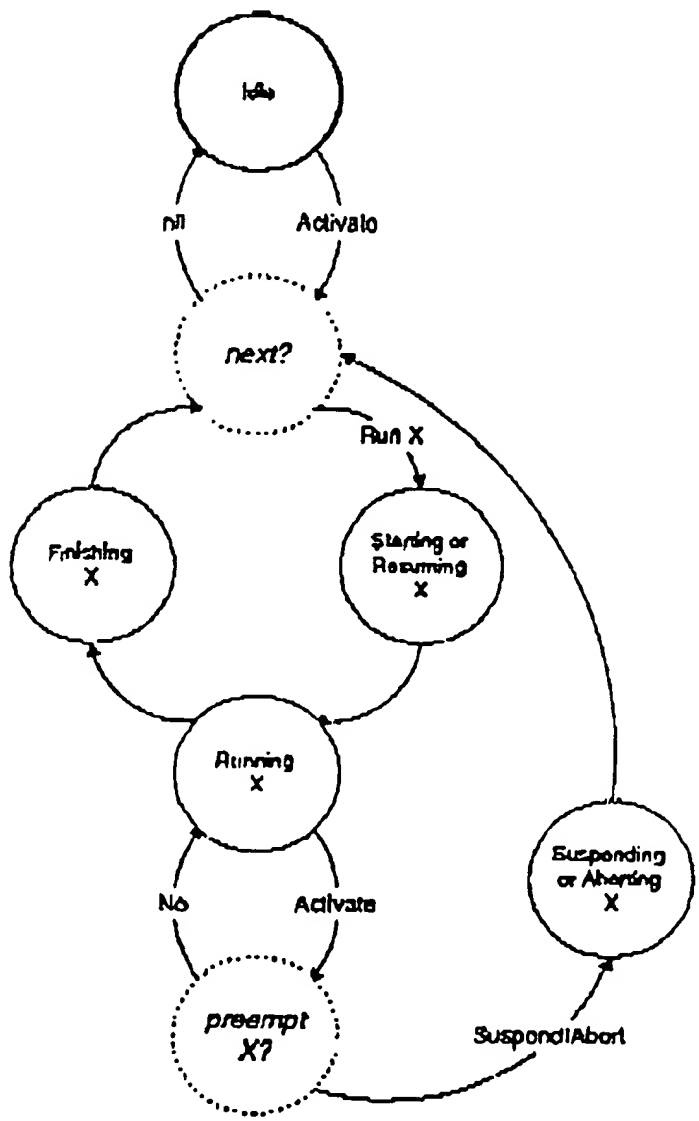
Idle
nil
Activate
next?
Run X
Finishing X
Starting or Resuming X
Running X
No
Activate
preempt X?
Suspend/Abort
Suspending or Aborting X

FIG. 7

```
task_ABD() {
        while (1) {             // cyclo-static schedule in task
                read_events();
                A_Run();        // 1
                B_Run();        // 2
                D_Run();        // 3
                post_events();
        }
}

irq_handler_C() {
        C_Run();                // interrupt handler
        post_events();
}

task_E() {
        while (1) {             // single behavior in task
                read_events();
                E_Run();
                post_events();
        }
}

main() {                        // initialization
        rtos_create_task(task_ABD, ...)
        rtos_install_handler(irq_handler_C, ...)
        rtos_create_task(task_E, ...);
        ...
}
```

FIG. 8